Модели операционных систем

Операционная система *– это сложная программа, в которой детали накладываются друг на друга. Чтобы система могла обеспечивать желаемые возможности, необходима унифицирующая модель.*

Рассмотрим подходы к проектированию операционных систем на примере распространенных моделей.

Модель операционной системы *можно представить как каркас, который связывает в единое целое все средства и сервисы, обеспечиваемые системой, с одной стороны, и выполняемые ею задачи с другой.*

Существует множество способов структурирования операционных систем.

модель

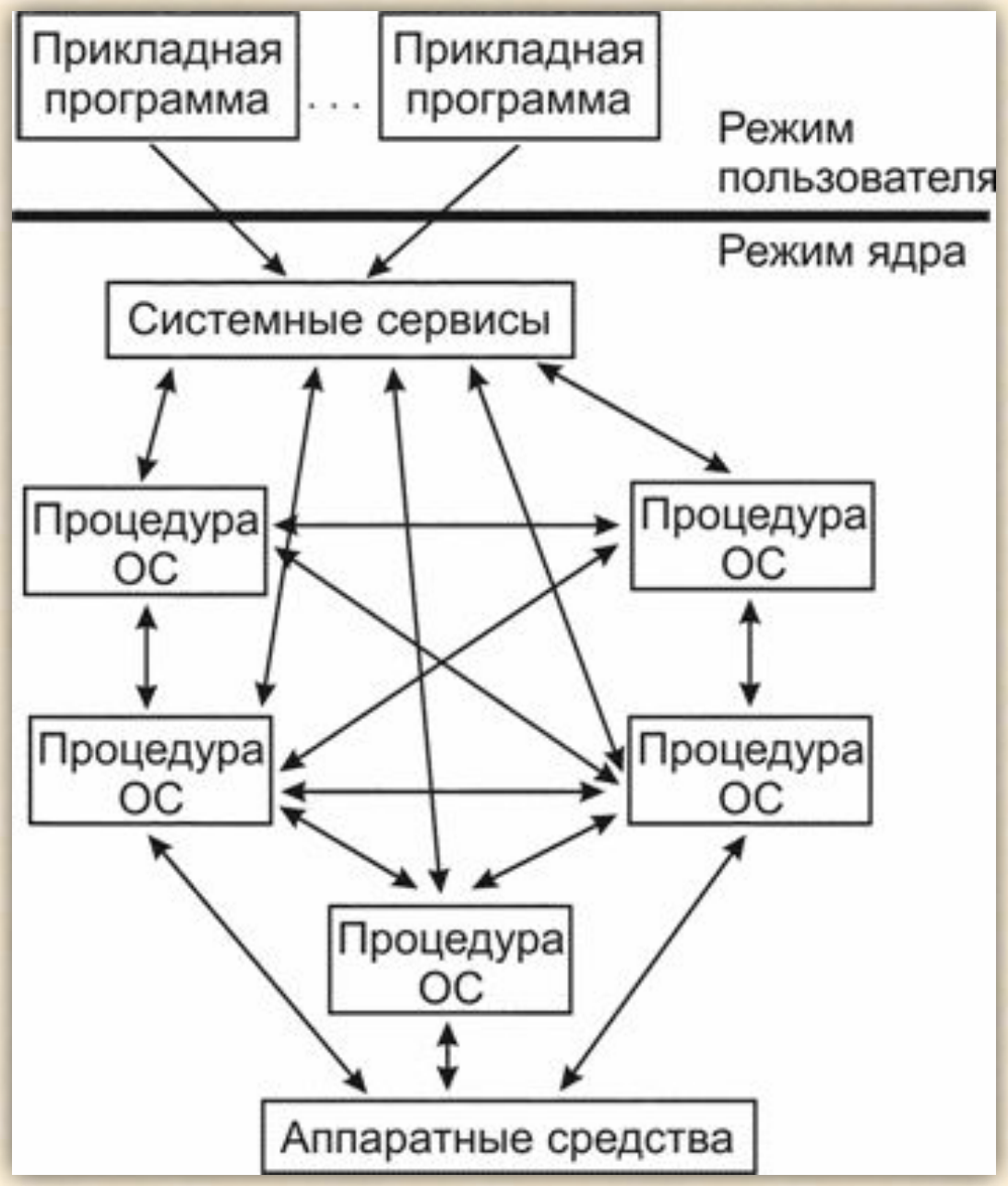
В небольших операционных системах, например, в MS-DOS, используют принцип организации системы, как набора процедур, каждую из которых может вызывать любая пользовательская процедура.

Такая модель называется монолитной

Недостатки:

1. *Такая структура не обеспечивает изоляции данных, поскольку в разных участках кода используется информация об устройстве всей системы.*

2. *Расширение операционных систем такого типа затруднительно, так как изменение некоторой процедуры может вызвать ошибки в других частях системы, на первый взгляд не имеющих к ней отношения.*



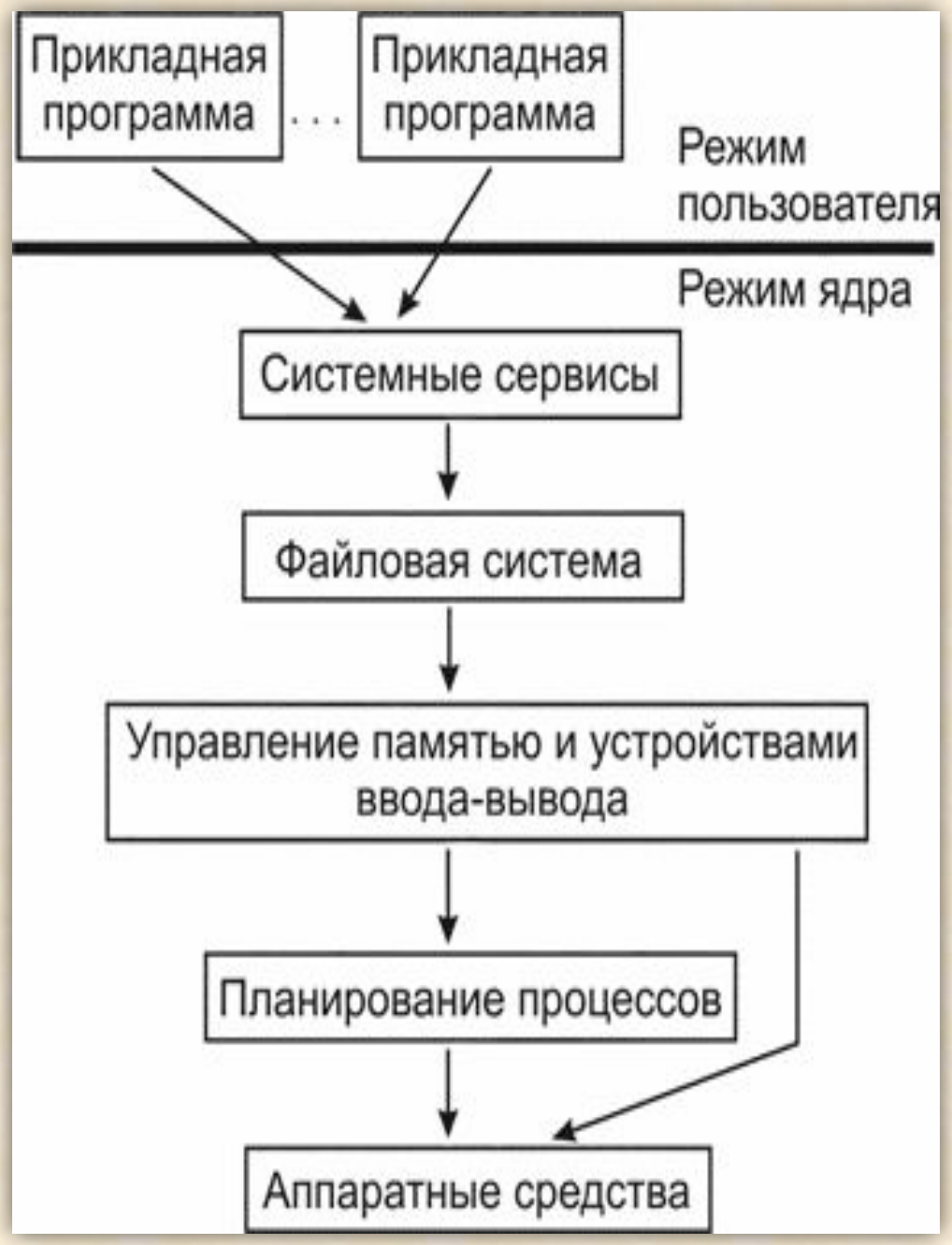
модель

Предполагает разделение системы на модули, наслоенные один поверх другого. Каждый модуль предоставляет набор функций, которые могут вызываться другими модулями. Код, расположенный в некотором слое, вызывает код только из нижележащих слоев.

В некоторых операционных системах, строящихся по данной модели, например, в VAX/VMS или в системе Multics, многослойность даже принудительно обуславливается аппаратными средствами.

Достоинства:

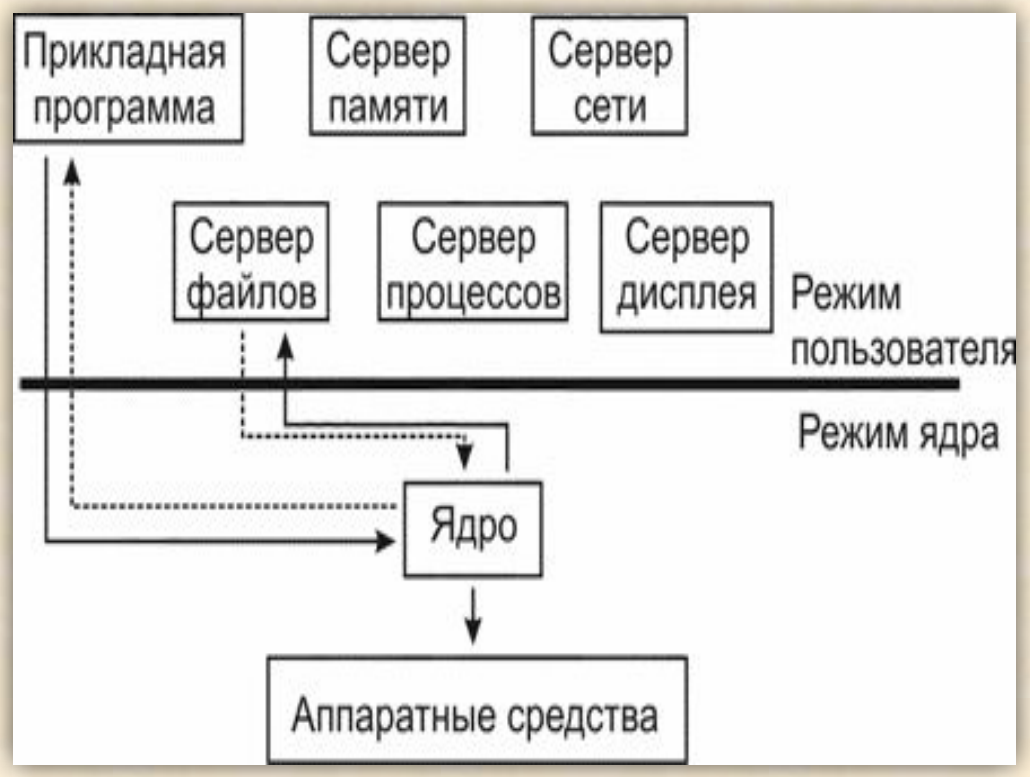
1. *Код каждого слоя получает доступ только к необходимым ему интерфейсам и структурам данных нижележащих слоев. Таким образом, уменьшается объем кода, обладающего неограниченной властью.*
2. *Такая структура позволяет при отладке операционной системы начинать с самого нижнего слоя и добавлять по одному уровню до тех пор, пока вся система не стает работать правильно.*
3. *Послойная структура облегчает и расширение систем, можно целиком заменить любой уровень, не затрагивая остальных частей.*



МОДЕЛЬ КЛИЕНТ-СЕРВЕР

Идея его состоит в разделении системы на несколько процессов, каждый из которых реализует один набор сервисов: например, распределение памяти, создание процессов или планирование процессов.

Каждый сервер выполняется в режиме пользователя, все время проверяя, не обратился ли к нему за обслуживанием какой-либо клиент. Клиент, которым может быть либо другой компонент операционной системы, либо прикладная программа, запрашивает выполнение сервиса, посылая серверу сообщение. Ядро операционной системы, выполняющееся в режиме ядра, доставляет сообщение серверу. Тот выполняет запрашиваемые действия, после чего ядро возвращает клиенту результаты в виде другого сообщения.



Достоинства:

1. *Операционная система, состоящая из автономных компонентов имеет небольшой размера.*
2. *Поскольку все серверы выполняются как отдельные процессы в режиме пользователя, авария и, возможно, перезапуск одного из серверов не нарушает работы остальных частей системы.*
3. *Разные серверы могут выполняться на разных процессорах многопроцессорного компьютера или даже на разных компьютерах. Это делает операционную систему, построенную по такой модели пригодной для распределенных*

Объектная модель

Как и в случае других больших программных систем, трудно найти одну, главную программу, которая управляет всей операционной системой. Таким образом, вместо того, чтобы разрабатывать систему сверху вниз, по объектно-ориентированной методологии сначала рассматривают данные, с которыми должна работать программа для выполнения своей задачи.

Для операционных систем такими данными являются системные ресурсы – файлы, процессы, память и так далее.

Основная цель разработки системы с ориентацией на данные – создание программного обеспечения, которое можно было бы легко, а главное дешево, изменять. Один из способов минимизации необходимых изменений в объектно-ориентированных программах – это сокрытие физического представления данных внутри объектов. Объект представляет собой структуру данных, физический формат которой скрыт в определении типа. Он имеет набор свойств, называемых атрибутами, и с ним работает группа сервисов.

Многие операционные системы используют объекты для представления системных ресурсов. Каждый системный ресурс, который могут совместно использовать несколько процессов, реализован как объект и обрабатывается объектными сервисами. Такой подход уменьшает влияние изменений, которые могут происходить в системе с течением времени. Например, если изменение в операционной системе вызвано изменением в аппаратуре, необходимо изменить только объект, представляющий данный аппаратный ресурс, и его сервисы. При этом код, который использует объект, не нуждается в модификации. Аналогично, если нужно ввести в систему поддержку нового устройства, создается новый объект, и добавление его к системе не нарушает существующего кода.

Помимо того, что уменьшается влияние изменений, построение операционной системы на основе объектов имеет еще ряд несомненных ***преимуществ:***

- *- Доступ системы к ресурсам и работа с ними унифицированы. Создание, удаление и ссылка на объект осуществляется совершенно аналогично. И поскольку каждый ресурс – это объект, контроль использования ресурсов сводится к отслеживанию создания и использования объектов.*
- *- Упрощается защита, так как для всех объектов она осуществляется одинаково. При попытке доступа к объекту подсистема защиты вмешивается и проверяет допустимость операции, независимо от того, какой ресурс представляет объект.*
- *- Объекты предоставляют удобную и унифицированную базу для совместного использования ресурсов двумя или несколькими процессами. Для работы с объектами любого типа используются описатели объектов. Два процесса совместно используют объект тогда, когда каждый из них открыл его описатель. Система может отслеживать количество описателей, открытых для данного объекта, чтобы определить, действительно ли он все еще используется. После этого система может уда лить объекты, которые более не используются.*